UNIVERSIDAD DE LA RIOJA

TRABAJO FIN DE ESTUDIOS

Título

Aplicación de soporte a prácticas en laboratorios

Autor/es

CLARA GARCÍA VIGUERA

Director/es

ÁNGEL LUIS RUBIO GARCÍA y JUDITH MILLÁN MONEO

Facultad

Facultad de Ciencia y Tecnología

Titulación

Grado en Ingeniería Informática

Departamento

MATEMÁTICAS Y COMPUTACIÓN

Curso académico

2016-17

publicaciones.unirioja.es
E-mail: publicaciones@unirioja.es

UNIVERSIDAD DE LA RIOJA

Facultad de Ciencia y Tecnología

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

APLICACIÓN DE SOPORTE A PRÁCTICAS EN LABORATORIOS

Alumno:

Clara García Viguera

Tutores:

Ángel Luis Rubio García

Judith Millán Moneo

Logroño, Febrero, 2017

Tabla de contenido

Tabla de Figuras

Resumen

El objetivo de este proyecto es realizar una aplicación web que sirva de soporte al personal docente de la Universidad de La Rioja en la gestión de las prácticas en laboratorio. Esta aplicación busca mejorar la coordinación entre las sesiones de laboratorio de las diferentes asignaturas de la titulación de grado.
Como ejemplo concreto de utilización, se va a aplicar la herramienta para el Grado en Química, titulación para la que se ha desarrollado un proyecto de innovación docente que analiza esta coordinación.

Abstract

The aim of it consists of designing a web application to support the teaching staff of La Rioja University in the management of laboratory practices. The main purpose of this application is to provide a better coordination between each laboratory session at the different subjects of the degree.
As a particular example, the application will be applied for the degree in Chemistry, in which a teaching innovations project that analyzes this coordination has recently been developed.

Capítulo 1

Introducción

En este capítulo se introducirá el proyecto y su contexto, así como los objetivos iniciales. También se detallarán las tecnologías a usar para el desarrollo del mismo.

1. Introducción

1.1. Contexto

La realización de este proyecto surge de la necesidad de coordinar tanto horizontal como transversalmente las diferentes prácticas realizadas en los laboratorios o talleres (prácticas de informática, física, biología u otras) de los diferentes grados universitarios. Dada la cantidad de información y lo variada que es, sería imposible pensar en gestionar esta información de forma manual. Para solucionarlo se plantea la creación de un repositorio o gestor documental de las diferentes prácticas de un determinado grado, de manera que sea más sencillo organizarlas y coordinarlas. Como caso particular vamos a considerar el grado en Química, en el cual esta idea ha dado lugar a un proyecto de innovación docente que hace que la propuesta esté más avanzada.

1.2. Objetivos iniciales

El objetivo de este gestor documental será archivar la lista de prácticas por sus títulos, acompañados de una pequeña descripción y etiquetadas mediante una serie de palabras clave que indiquen el contenido trabajado y las técnicas aplicadas. Además estas prácticas estarán organizadas por módulos, materias, cursos, semestres y asignaturas. La aplicación está dirigida a los profesores responsables de las asignaturas de grado. Este gestor deberá dar soporte a la realización de consultas mediante palabras clave, además de realizar modificaciones en las prácticas o añadir nuevas prácticas al gestor. Otro objetivo del proyecto es integrar la aplicación en la infraestructura de la Universidad, integrando de forma particular el sistema de autenticación de usuario que se utiliza en las distintas plataformas informáticas.

Para llevar a cabo este proyecto, en particular para el grado de Química, dispongo de un documento en el cual se detallan las diferentes prácticas del primer y segundo curso, con su descripción y palabras clave asociadas, además de un gráfico con las relaciones entre las diferentes asignaturas de la titulación. También se cuenta con un diseño inicial de base de datos y con la ayuda de Judith Millán, coordinadora del proyecto de innovación docente de grado de Química, y mi tutor, Ángel Luis Rubio, para cualquier duda que me pueda surgir a lo largo del desarrollo del proyecto.

1.3. Tecnologías

Al comienzo del proyecto no se había establecido ninguna tecnología ni ningún lenguaje en concreto a utilizar, por lo que dada esta libertad me pareció interesante utilizar PHP como lenguaje de programación, ya que nunca antes lo había utilizado y se adaptaba bastante bien al proyecto. Esto supone un hándicap al ser algo nuevo y desconocido, pero me pareció una experiencia muy enriquecedora.

Además de PHP como lenguaje de programación se estudiará el uso de algún framework que sirva de ayuda a la hora de implementar la aplicación web. El estudio queda reflejado en el anexo 3.

Capítulo 2

Planificación

En este capítulo se detallará la planificación del proyecto. Desde el número de horas a invertir y su distribución temporal, hasta el desglose de las tareas a llevar a cabo y su estimación temporal.

2. Planificación

Dado mi caso particular, las horas que se emplearán en el desarrollo del proyecto quedarán repartidas de la siguiente forma:

- Hasta el mes de diciembre solo podré dedicar unas 3 horas diarias al proyecto, ya que hasta esa fecha por las mañanas estaré realizando las prácticas en empresa.
- A partir de diciembre, en el momento que acabe las prácticas, estimo unas 7 horas de trabajo diario en el proyecto.

El calendario de trabajo sería algo similar a lo siguiente:

	NOVIEMBRE						
	L	M	X	J	V	S	D
		1	2	3	4	5	6
1	7	8	9	10	11	12	13
2	14	15	16	17	18	19	20
3	21	22	23	24	25	26	27
4	28	29	30				

	DICIEMBRE						
	L	M	X	J	V	S	D
4				1	2	3	4
5	5	6	7	8	9	10	11
6	12	13	14	15	16	17	18
7	19	20	21	22	23	24	25
8	26	27	28	29	30	31	

	ENERO						
	L	M	X	J	V	S	D
8							1
9	2	3	4	5	6	7	8
10	9	10	11	12	13	14	15
11	16	17	18	19	20	21	22
12	23	24	25	26	27	28	29
13	30	31					

	FEBRERO						
	L	M	X	J	V	S	D
13			1	2	3	4	5
14	6	7	8	9	10	11	12
15	13	14	15	16	17	18	19
	20	21	22	23	24	25	26
	27	28					

Inicio del proyecto.
3 horas diarias de trabajo aproximadas.
7 horas diarias de trabajo aproximadas.
Deposito del TFG.

Figura 1. Calendario de trabajo.

De esta manera son 29 días de 3 horas de trabajo, que equivale a 87 horas de trabajo más 35 días de 7 horas de trabajo lo que equivale a 245 horas de trabajo, haciendo un total de 332 horas de posible trabajo, que se entiende que está sujeto a pequeñas variaciones puntuales de días y horas realizadas.

2.1. Estructura de Descomposición del Trabajo (EDT)

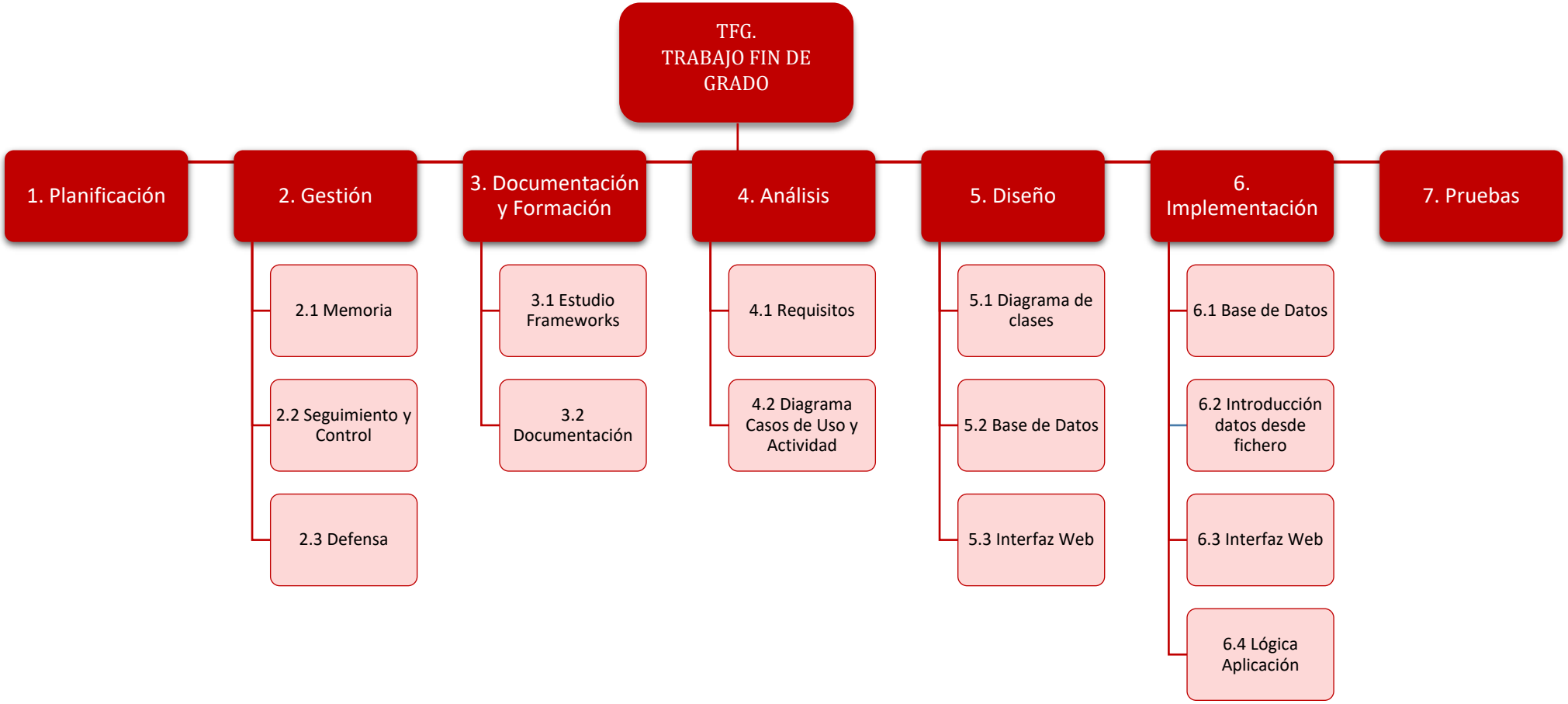


Figura 2. Estructura de Descomposición del Trabajo (EDT).

2.2. Descripción de las tareas

Las tareas a realizar y las horas estimadas quedan detalladas en la siguiente tabla:

Código	Tarea	Descripción	Horas estimadas
1	Planificación	Planificación del proyecto y las tareas a realizar. Descomposición en tareas (EDT) y distribución temporal (diagrama de Gantt).	10 h
2	Gestión	Tareas transversales a realizar en paralelo a la realización del proyecto.	55 h
2.1	Elaboración de la memoria	Realización de la memoria del proyecto.	30 h
2.2	Seguimiento y control	Reuniones tanto con el tutor, como con el "cliente" en este caso la coordinadora del proyecto de innovación docente, Judith Millán, control de riesgos y control de calidad.	20 h
2.3	Defensa	Elaboración de la presentación y preparación de esta para su defensa	15 h
3	Documentación y formación	Documentación y formación sobre las tecnologías a usar y la utilización o no de un framework.	45 h
3.1	Estudio frameworks	Estudio de un número determinado de frameworks que puedan adaptarse a las necesidades del proyecto.	15 h
3.2	Documentación	Documentación acerca de las diferentes tecnologías que se usaran en el proyecto, principalmente PHP, al ser desconocido.	30 h
4	Análisis	Obtención de los requisitos tanto funcionales como no funcionales que debe cumplir la aplicación web.	10 h
4.1	Requisitos	Obtención de los requisitos funcionales y no funcionales que debe cumplir la aplicación.	4 h
4.2	Diagrama casos de uso y de actividad	Realización del diagrama de casos de uso y diagramas de actividad más destacados.	6 h
5	Diseño	Diseño de la interfaz web, base de datos y lógica de la aplicación.	10 h

5.1	Diagrama de clases	Diagrama que describe la estructura de la aplicación.	4 h
5.2	Base de datos	Diseño de la base de datos para que recoja toda la información necesaria para la aplicación.	4 h
5.3	Interfaz	Diseño de os prototipos para la interfaz de la aplicación.	2 h
6	Implementación	Implementación de la aplicación web, presentación, lógica y persistencia.	165 h
6.1	Base de datos	Creación de la base de datos y la conexiones a ella.	30 h
6.2	Introducción datos desde fichero	Introducción de los datos proporcionados en el documento.	15 h
6.3	Interfaz Web	Implementación de toda la capa de presentación de la aplicación.	55 h
6.4	Lógica de la aplicación	Implementación de la lógica de la aplicación a la hora de buscar las prácticas y mostrarlas, filtros y ordenación.	65 h
7	Pruebas	Fase de pruebas en busca de posibles fallos o inconsistencias en la aplicación.	15 h
	Total		**310 h**

Figura 3. Tareas y horas estimadas.

2.3. Diagramas de Gantt

He dividido el diagrama de Gantt para mejor visibilidad aprovechando los diferentes periodos de mi proyecto, el primero con unas 3 horas de trabajo diario y un segundo periodo con 7 horas de trabajo diario aproximado. Las tareas de memoria, seguimiento y control se desarrollaran a lo largo de todo el proyecto. A pesar de que hay 332 horas disponibles solo se han planificado 310. Esta diferencia ha sido pensada especialmente para poder aprovecharla en el caso de encontrar imprevistos, desvíos o modificaciones.

	Activity	Inicio	Final		Días	Horas
1	**Planificacion**	**15-11-16**	**21-11-16**	**4.5**	**4.5**	**10**
2	**Gestión**	**07-11-16**	**15-02-17**		**143.0**	**55**
2.1	Memoria	07-11-16	13-02-17	70.5		30
2.2	Seguimiento y control	07-11-16	13-02-17	70.5		20
2.3	Defensa	14-02-17	15-02-17	2.0		15
3	**Documentación y formación**	**23-11-16**	**07-02-17**		**60.5**	**45**
3.1	Estudio framewroks	29-11-16	07-12-16	6.0		15
3.2	Documentación	23-11-16	07-02-17	54.5		30
4	**Análisis**	**18-11-16**	**29-11-16**		**7.0**	**10**
4.1	Requisitos	18-11-16	23-11-16	3.0		4
4.2	Diagramas CdU y Actividad	23-11-16	29-11-16	4.0		6
5	**Diseño**	**08-12-16**	**20-12-16**		**8.0**	**10**
5.1	Clases	08-12-16	13-12-16	3.0		4
5.2	Base de Datos	13-12-16	16-12-16	3.0		4
5.3	Interfaz	16-12-16	20-12-16	2.0		2
6	**Implementación**	**21-12-16**	**02-02-17**		**31.5**	**165**
6.1	Base de Datos	21-12-16	29-12-16	7.0		30
6.2	Intro. datos fichero	30-12-16	05-01-17	4.5		15
6.3	Interfaz web	05-01-17	18-01-17	9.5		55
6.4	Lógica de la aplicación	19-01-17	02-02-17	10.5		65
7	**Pruebas**	**02-02-17**	**07-02-17**	**3.5**	**3.5**	**15**

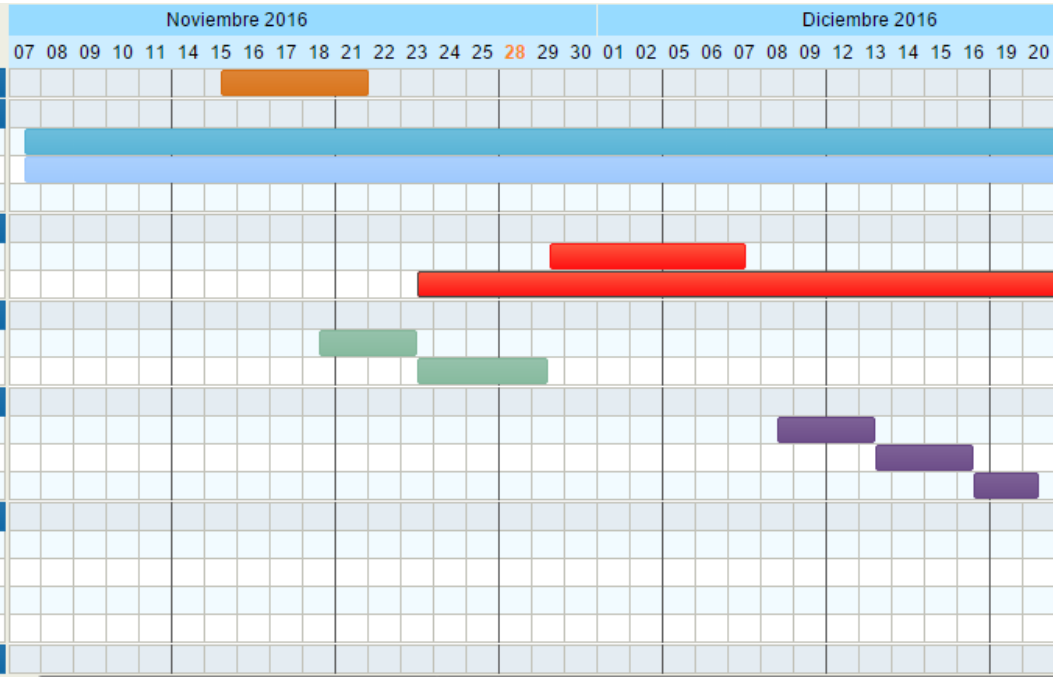
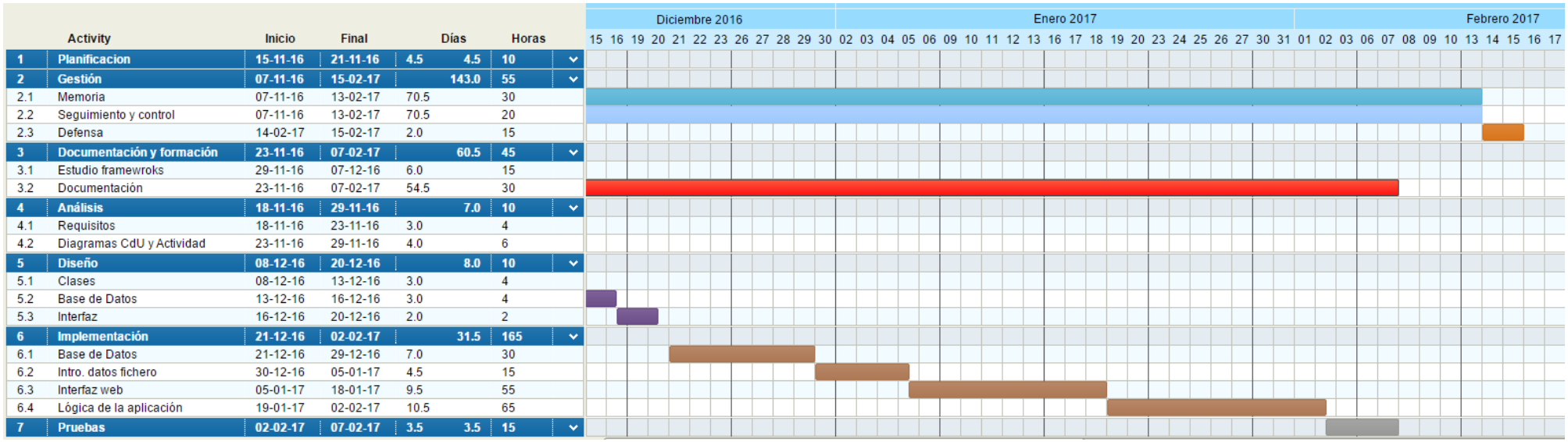


Figura 4. Diagrama de Gantt (1).



	Activity	Inicio	Final	Días		Horas
1	Planificacion	15-11-16	21-11-16	4.5	4.5	10
2	Gestión	07-11-16	15-02-17		143.0	55
2.1	Memoria	07-11-16	13-02-17	70.5		30
2.2	Seguimiento y control	07-11-16	13-02-17	70.5		20
2.3	Defensa	14-02-17	15-02-17	2.0		15
3	Documentación y formación	23-11-16	07-02-17		60.5	45
3.1	Estudio framewroks	29-11-16	07-12-16	6.0		15
3.2	Documentación	23-11-16	07-02-17	54.5		30
4	Análisis	18-11-16	29-11-16		7.0	10
4.1	Requisitos	18-11-16	23-11-16	3.0		4
4.2	Diagramas CdU y Actividad	23-11-16	29-11-16	4.0		6
5	Diseño	08-12-16	20-12-16		8.0	10
5.1	Clases	08-12-16	13-12-16	3.0		4
5.2	Base de Datos	13-12-16	16-12-16	3.0		4
5.3	Interfaz	16-12-16	20-12-16	2.0		2
6	Implementación	21-12-16	02-02-17		31.5	165
6.1	Base de Datos	21-12-16	29-12-16	7.0		30
6.2	Intro. datos fichero	30-12-16	05-01-17	4.5		15
6.3	Interfaz web	05-01-17	18-01-17	9.5		55
6.4	Lógica de la aplicación	19-01-17	02-02-17	10.5		65
7	Pruebas	02-02-17	07-02-17	3.5	3.5	15

Figura 5. Diagrama de Gantt (2).

2.4. Diagrama de Hitos

	Semanas															
Hitos	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16-17
Comienzo TFG	■															
Reuniones con el tutor	■	■			■			■			■			■		
Entrega introducción y planificación		■														
Entrega análisis				■												
Comienzo estudio frameworks				■												
Comienzo diseño					■											
Fin diseño						■										
Comienzo implementación							■									
Fin implementación												■				
Comienzo pruebas												■				
Deposito TFG															■	
Defensa																■

Figura 6. Diagrama de hitos.

2.5. Plan de comunicación

A lo largo del desarrollo del proyecto, será necesario llevar a cabo una buena comunicación con Judith Millán, en el papel de cliente del proyecto y con Ángel Luis Rubio, como tutor del proyecto. Para ello hemos establecido distintos medios de comunicación.

- Correo electrónico: tanto con Judith como con Ángel Luis, será el primer medio de comunicación, tanto para informar de la evolución del proyecto o imprevistos, como para establecer fechas de reunión.
- Reuniones: se acordará con el tutor del TFG una reunión cada 3 semanas aproximadamente, y en cualquier circunstancia en la que se necesite, acordando fecha y hora por el medio de comunicación anterior. Todas las actas de las reuniones mantenidas a lo largo del desarrollo del proyecto quedarán recogidas en el anexo 1.

2.6. Plan de riesgos

A lo largo del desarrollo del proyecto es bastante probable que nos encontremos con problemas y dificultades, por lo que detectarlas a tiempo es importante para poder corregirlas y que estas no provoquen un desvío demasiado significativo en la evolución del proyecto. A continuación detallo alguno de los riesgos que puedo encontrar a lo largo del proyecto.

- *Perdida de información:* para evitar lo máximo posibles retrasos por esta causa, se realizarán copias cada semana y se guardarán las dos últimas. Además para el código se utilizará un control de versiones de manera que el código siempre esté disponible.

- *Mala planificación:* se llevará un seguimiento del proyecto semanalmente y en el caso de detectar algún desvío significativo se tendrá en cuenta para corregirlo.
- *Desconocimiento de las tecnologías:* es un riesgo bastante probable por elegir PHP como lenguaje de programación y ser nuevo para mí. Para poder minimizar el impacto de este riesgo se planifican en el desarrollo del proyecto 30 horas de documentación y 15 horas para un estudio de frameworks que puedan hacer más amigable el trabajo con un lenguaje desconocido.
- *Cambios en los requisitos y alcance:* para minimizar al máximo posible este riesgo, se dejarán fijados claramente al inicio del proyecto, y se mostrará al cliente lo antes posible una versión final del producto.

Capítulo 3

Análisis

Este capítulo está dedicado a recoger los requisitos que ha de cumplir nuestra aplicación, los diferentes roles que aparecerán, diagrama de casos de uso y algún diagrama de actividad.

3. Análisis

3.1. Requisitos de la aplicación

La aplicación deberá almacenar todas las sesiones prácticas (laboratorios o informáticas) de las asignaturas de cada grado. De cada una de ellas se almacenará un título, descripción y palabras clave asociadas.
Además de las prácticas se quiere guardar la asignatura a la que pertenece cada práctica, la titulación a la que pertenece y el profesor o profesores que la imparten. Cada asignatura estará identificada por un código, y tendrá nombre, modulo, materia curso y semestre al que pertenece.
También se quiere recoger las relaciones que hay entre las diferentes asignaturas de una misma titulación, y por lo tanto entre sus prácticas.
Además también debe ser posible a través de la aplicación de gestionar las asignaturas (añadir, modificar o eliminar), titulaciones (añadir, modificar o eliminar) y gestión de usuarios (añadir y eliminar). Las prácticas podrán ser modificadas y añadidas, pero nunca eliminadas.
El principal objetivo de la aplicación es poder buscar las relaciones entre las prácticas de distintas asignatura para así poder coordinar lo mejor posible todas las asignaturas y sus prácticas. La aplicación permitirá buscar y filtrar las prácticas almacenadas mediante palabras clave, módulos, materias, cursos, semestres o una combinación de ambas.
Para acceder a la aplicación será necesario un identificador, que será la CUASI, y una contraseña.
Se han identificado cuatro tipos de usuario o roles, del más básico, usuario básico, hasta el superadministrador, pasando por usuario intermedio y administrador.

- ✓ El usuario básico podrá registrarse, iniciar sesión y realizar consultas.
- ✓ El usuario intermedio también podrá añadir una nueva práctica o modificar una ya existente. El administrador podrá además gestionar las asignaturas, añadiéndolas o modificándolas.
- ✓ El superadministrador podrá realizar cualquiera de las anteriores acciones además de gestionar los usuarios, añadiendo o eliminando y gestionar las titulaciones, añadiendo o modificando. En principio eliminar no estaba contemplado, pero en caso de que fuera necesario será este rol quien tenga privilegios para realizarlo.

Un usuario básico podrá ser cualquier profesor de la titulación, mientras que usuario avanzado serán únicamente los profesores responsables de alguna asignatura. El administrador será aquel que es director de estudios y por último el superadministrador será algún responsable académico de la Facultad, o la persona encargada de administrar la aplicación en su defecto.

Requisitos no funcionales

- ✓ La aplicación estará adaptada y optimizada a los principales navegadores web: Chrome y Firefox.
- ✓ La aplicación será lo suficientemente segura para impedir ataques o poner en riesgo los datos almacenados.
- ✓ Pese a que la funcionalidad de la aplicación es sencilla, dados los requisitos de la misma, era indispensable contar con una aplicación web.

- ✓ Otro requisito es alojar la aplicación en un servidor dentro de la Universidad, al que los profesores pudieran acceder para consultar cuando quisieran.
- ✓ Se buscará utilizar el sistema de autenticación de la Universidad, por lo que para estos dos últimos puntos será necesario coordinarse y contar con la ayuda del servicio informático.

3.2. Diagrama casos de uso

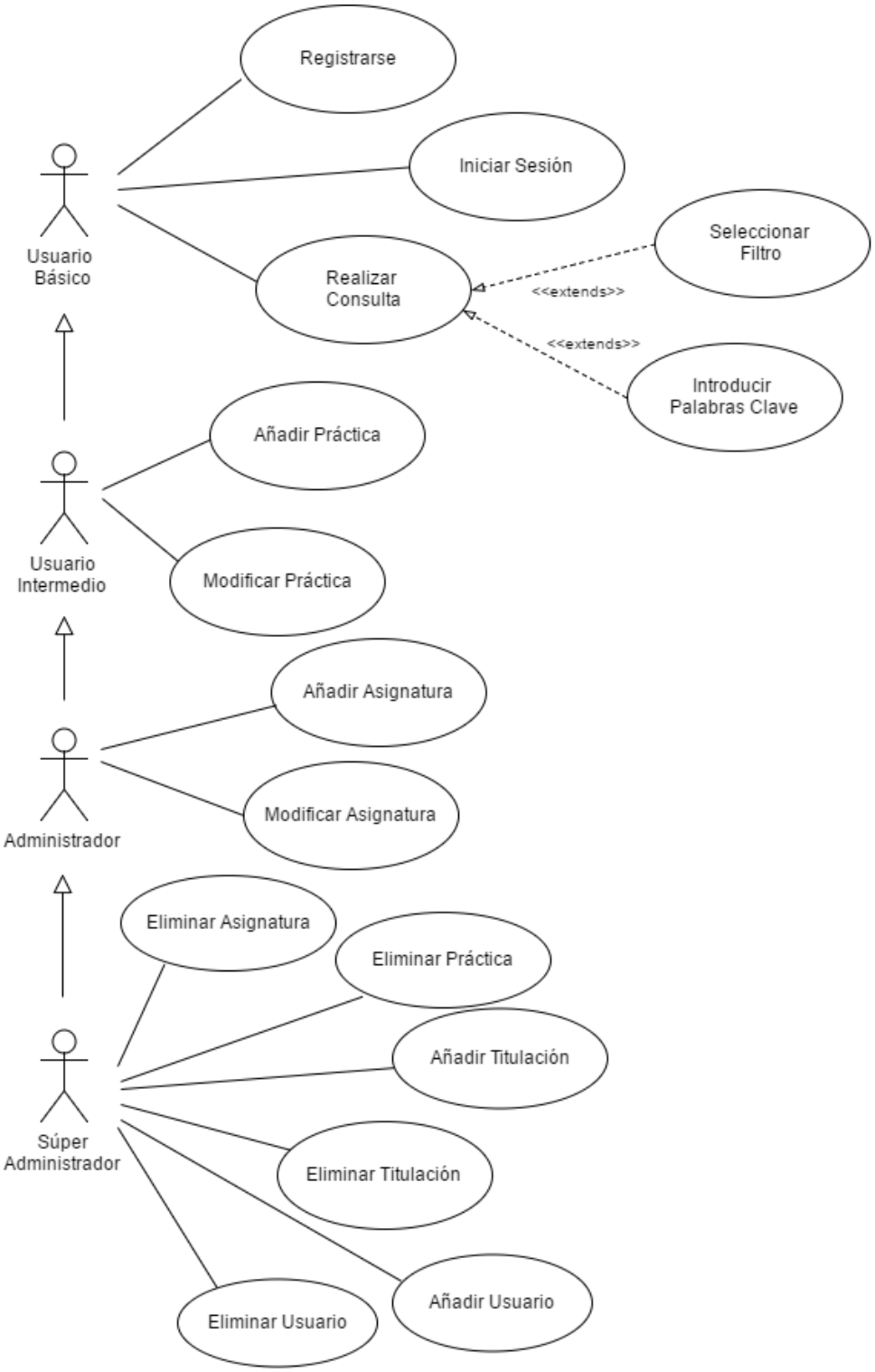


Figura 7. Diagrama de casos de uso.

3.3. Especificación casos de uso

La especificación de cada caso de uso del diagrama anterior queda reflejada en el Anexo 2, especificación de casos de uso.

3.4. Diagramas de actividad

A continuación, mediante los siguientes diagramas de actividad, detallarle alguno de los casos de uso anteriores.

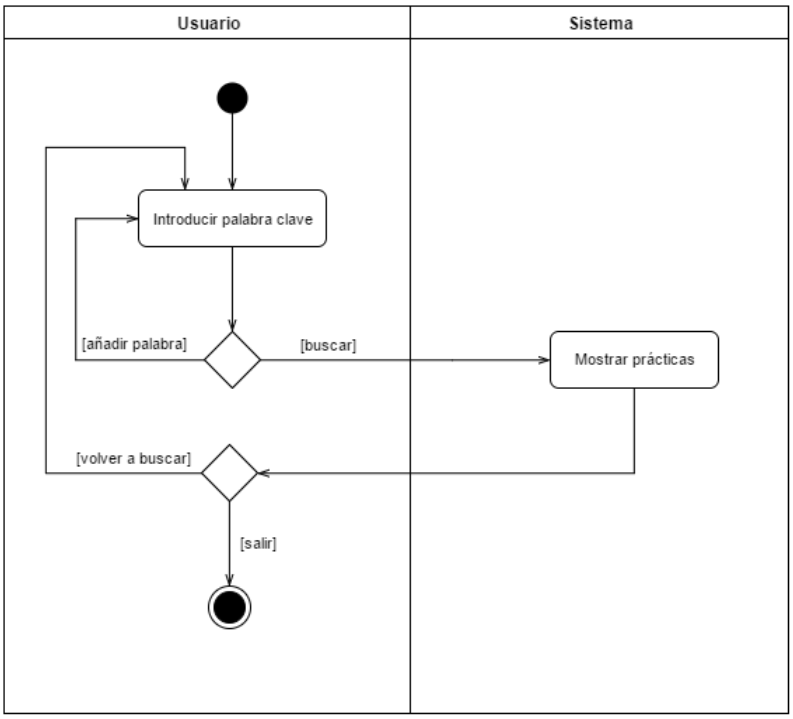


Figura 8. Diagrama actividad caso de uso "Realizar consulta".

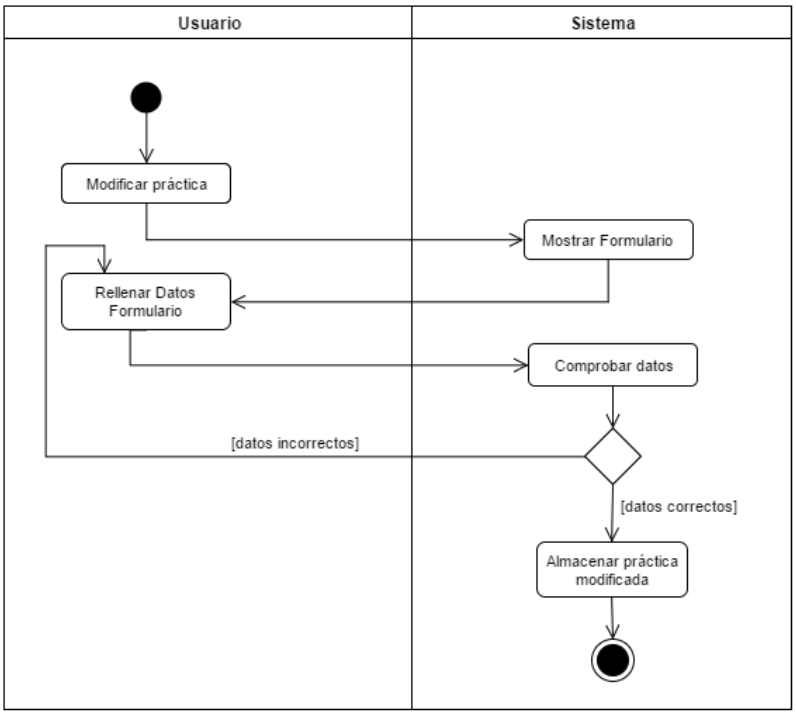


Figura 9. Diagrama de actividad caso de uso "Modificar práctica".

3.5. Modificación diagrama casos de uso.

Durante la realización del proyecto, se decidió añadir un par de casos de uso no contemplados al comienzo y que a continuación detallaré.
La idea inicial era que los usuarios de la aplicación accedieran a esta mediante el uso del sistema de autenticación de la Universidad de La Rioja, pero a medida que avanzada el proyecto esta opción no era posible llevarla a la realidad, dado que le tiempo avanzaba y los datos para realizar la conexión no habían sido facilitados.
Entonces se planteó un sistema de autenticación de usuarios de forma local, de tal manera que se crearía un grupo de usuarios a partir de los incluidos en el fichero facilitado por Judith Millán. Estos usuarios deberían poder modificar sus datos personales (usuario y contraseña) dotándoles de mayor comodidad de uso. Además será necesario establecer un sistema de notificaciones (seguramente mediante un correo electrónico), ya que los nuevos usuarios serán creados por el superadministrador. Estas modificaciones dieron lugar a los tres casos de uso que a continuación aparecen.

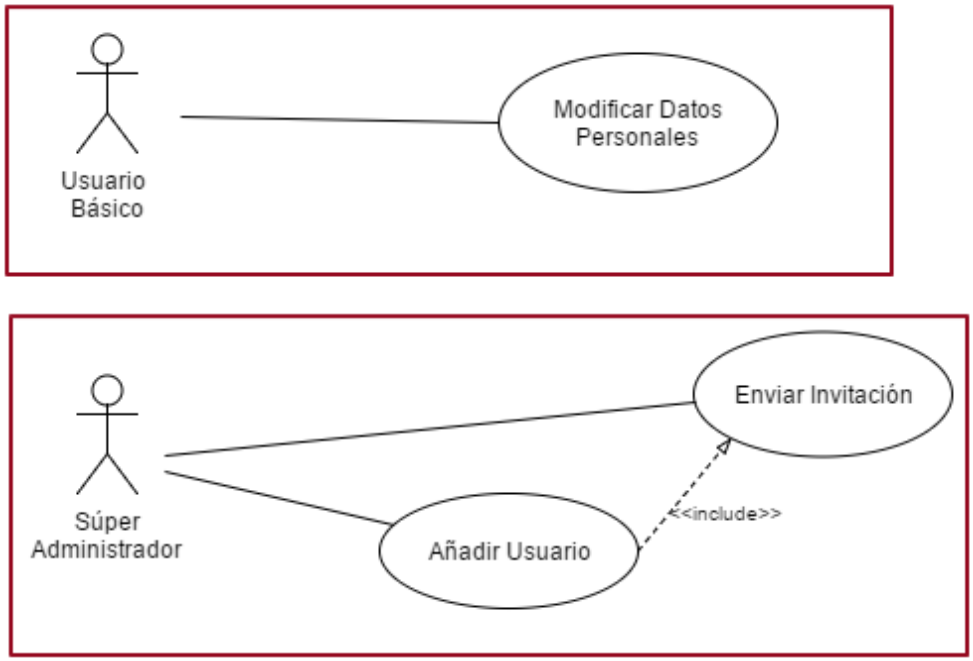


Figura 10. Casos de uso añadidos.

De esta manera el caso de uso registrarse del usuario básico, únicamente guardaría los datos del usuario, siendo el superadministrador quien le dé de alta en la aplicación y le envié la invitación de acceso.

Caso de Uso	Modificar datos personales
Descripción	El usuario de la aplicación modifica sus datos personales.
Actor	Usuario básico
Precondición	El usuario debe haber iniciado sesión previamente.
Pasos	• El usuario desde el botón de ajustes situado en la pantalla principal elige la opción modificar datos personales. • El usuario podrá modificar su usuario y contraseña. • Una vez el usuario quede de acuerdo con la modificación, confirma el cambio.
Postcondición	Los datos del usuario son actualizados en la base de datos.

Caso de Uso	Enviar invitación
Descripción	El usuario de la aplicación modifica sus datos personales.
Actor	Súper Administrador
Precondición	El usuario debe haber iniciado sesión previamente.
Pasos	• El usuario desde el botón de ajustes situado en la pantalla principal seleccionará la opción mandar invitación. • El usuario introducirá el correo electrónico de la persona a la que desea invitar y lo confirmará.
Postcondición	La invitación es enviada al correo especificado.

Caso de Uso	Añadir usuario
Descripción	El usuario podrá dar de alta a un nuevo usuario en la base de datos, mediante un correo, usuario y contraseña.
Actor	Súper Administrador
Precondición	El usuario debe haber iniciado sesión previamente. El usuario a añadir no debe estar en la base de datos.
Pasos	• El usuario desde el botón de ajustes situado en la pantalla principal elige la opción añadir usuario. • A través de un formulario rellena los campos necesarios y confirma para añadir. • Por último, este proceso implícitamente mandará un email al usuario registrado con una invitación para acceder a la aplicación y modificar sus datos personales.
Postcondición	El usuario queda añadido a la base de datos y la invitación es enviada.

Capítulo 4

Diseño

Este capítulo está dedicado a recoger los diferentes aspectos del diseño de la aplicación. Se detallarán aspectos y características destacadas de la base de datos y de la interfaz.

4. Diseño

4.1. Diseño de la base de datos

4.1.1. Descripción

De cada asignatura guardaremos su código, que además será único y por lo tanto servirá para identificarlo, nombre, módulo, curso, semestre y materia a la que pertenece dicha asignatura. Cada asignatura podrá estar relacionada con ninguna o varias asignaturas.
Una asignatura también estará asociada con uno o varios profesores que la impartirán. De cada profesor guardaremos su identificador único, nombre y apellidos.
Cada asignatura tendrá relación con la titulación o titulaciones a las que pertenece, ya que se puede dar el caso de que una misma asignatura se curse en dos titulaciones distintas. De cada titulación guardaremos su código de identificación único y su nombre.
Cada asignatura tendrá como mínimo una sesión práctica, pudiendo tener más de una. De cada sesión práctica la base de datos almacenará su identificador, nombre y descripción, así como la relación de las palabras clave con las que dicha sesión queda etiquetada. Cada sesión estará etiquetada por al menos una palabra clave y cada sesión práctica podrá estar relacionada únicamente con una asignatura.
De cada palabra clave almacenaremos su identificador y el nombre.
En la base de datos también tendremos almacenados los usuarios de la aplicación, de los cuales almacenaremos su identificador, email, contraseña y rol. Cada usuario estará ligado a un profesor, pero no todos los profesores tienen porque ser usuarios de la aplicación.

4.1.2. Diagrama entidad relación

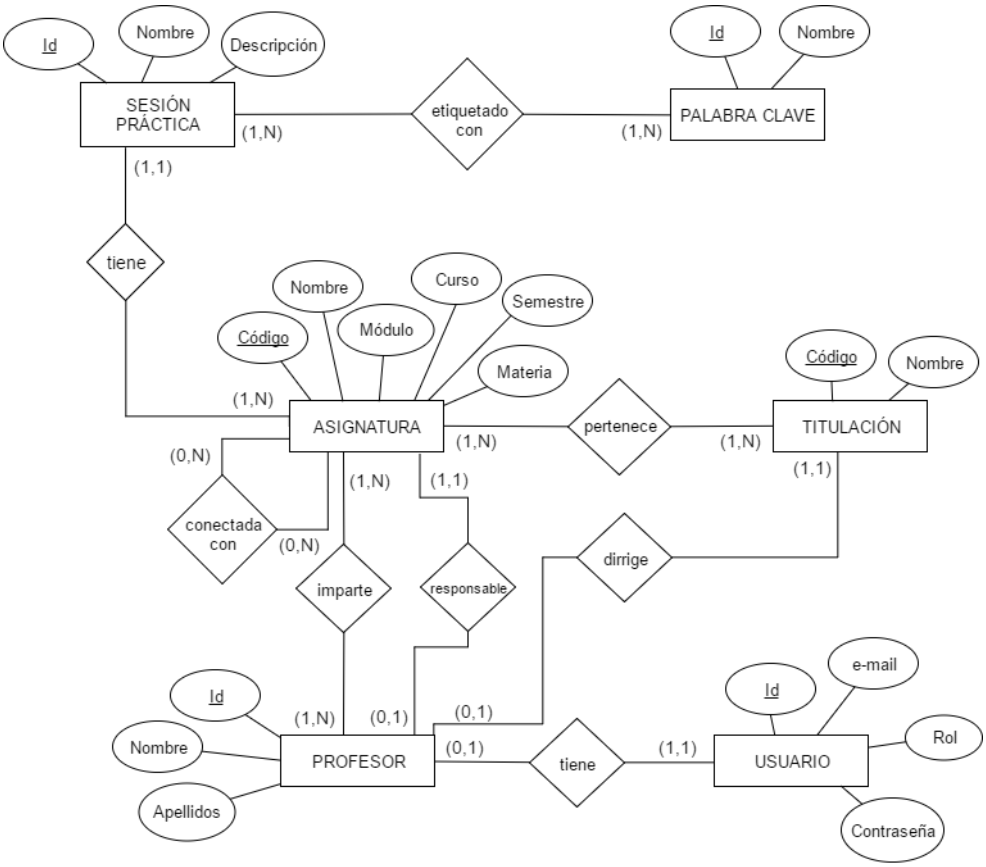


Figura 11. Diagrama ER.

4.1.3. Modelo relacional

ASIGNATURA

Código	Nombre	Módulo	Curso	Semestre	Materia	Responsable

CE: Profesor

Tipos de datos: código, nombre, módulo, semestre y materia (varchar). Curso (int).

TITULACIÓN

Código	Nombre	Director

CE: Profesor

Tipos de datos: código y nombre (varchar).

SESION_PRÁCTICA

Id	Nombre	Descripción	Asignatura

CE: Asignatura

Tipos de datos: id, nombre y descripción (varchar).

PALABRA_CLAVE

Id	Nombre

Tipos de datos: id (int). Nombre (varchar).

PROFESOR

Código	Nombre	Apellidos

Tipos de datos: código (int). Nombre y apellidos (varchar).

USUARIO

Código	E-mail	Contraseña	Rol	Código_Profesor

CE: Profesor

Tipos de datos: código, email, contraseña y rol (varchar).

ASIGNATURA_PROFESOR

Código_Asignatura	Código_Profesor

CE: Asignatura CE: Profesor

ASIGNATURA_TITULACIÓN

Código_Asignatura	Código_Titulación

CE: Asignatura CE: Titulación

ASIGNATURA_ASIGNATURA

Código_Asignatura	Código_Asignatura

CE: Asignatura CE: Asignatura

SESIÓN_PALABRAS_CLAVES

Código_Sesión	Id_Palabra

CE: Sesión CE: Palabra Clave

4.2. Diseño de la interfaz

Para el diseño de la interfaz se han creado unos prototipos de lo que se espera que sea visualmente la aplicación.

4.2.1. Iniciar sesión y registro

A continuación se muestra un prototipo de la interfaz de inicio de sesión en la aplicación. Una interfaz sencilla y común en los inicios de sesión. Constará de un logo de la Universidad de La Rioja, y debajo un formulario de login en el cual habrá que introducir usuario y contraseña de acceso a la aplicación. Aparecerá un enlace a registrarse, desde el cual un profesor no registrado podrá solicitar registro en la aplicación. Y por último el botón iniciar sesión el cual si los datos son correctos dará acceso a la aplicación.

Figura 12. Interfaz inicio de sesión.

4.2.2. Pantalla principal

A continuación muestro la pantalla principal que podrá visualizar un usuario básico de la aplicación, el cual podrá realizar las consultas. El usuario mediante un desplegable de opciones podrá seleccionar los filtros deseados y en el buscador introducir las palabras clave. Este buscador deberá mostrar las sesiones prácticas que coincidan con la búsqueda.

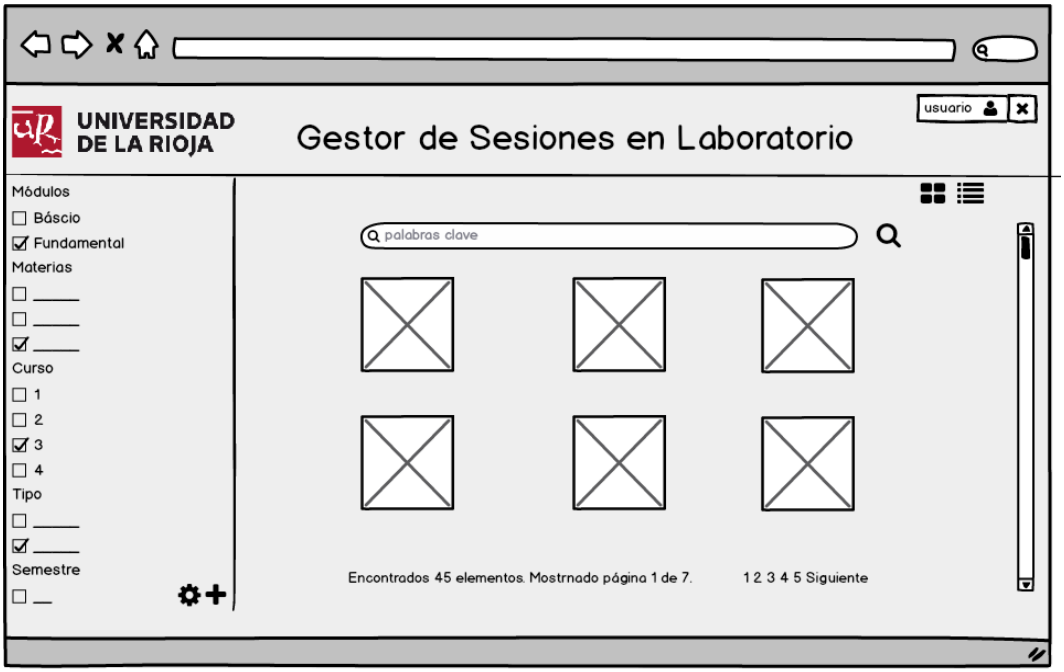


Figura 13. Interfaz página principal de la aplicación.

En el caso de que los usuarios puedan realizar más operaciones que la consulta, se mostrará un menú de la siguiente manera.

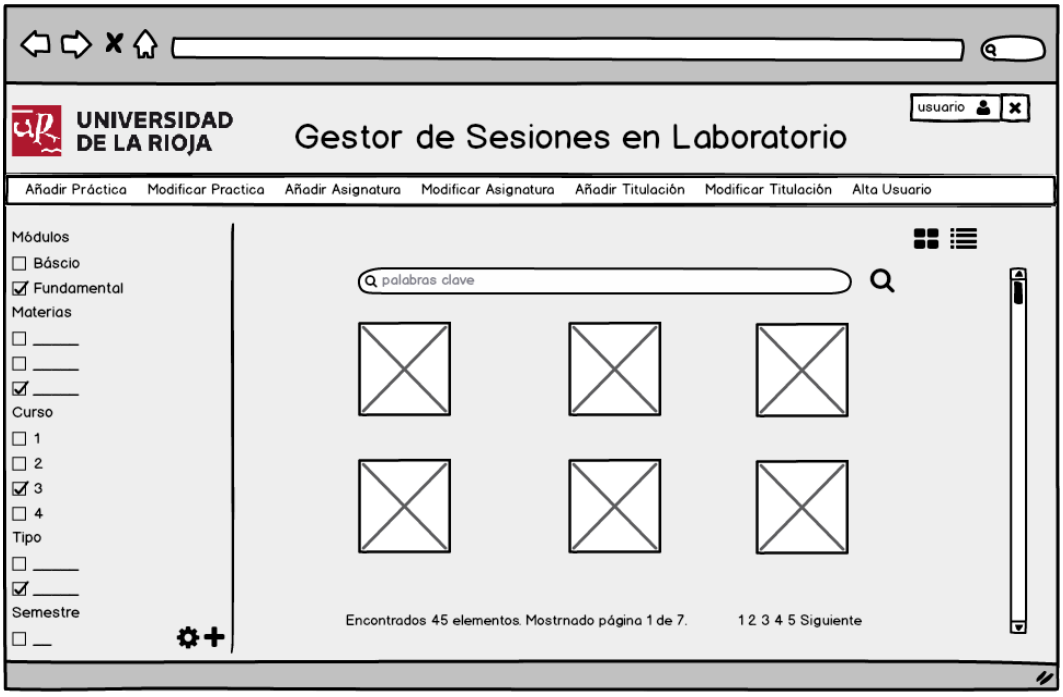


Figura 14. Interfaz página principal de la aplicación con menú.

Los elementos del menú variaran en función del rol del usuario y por tanto de las acciones que este puede realizar.
En la parte central aparecerán las prácticas, donde la vista se podrá modificar entre lista o mosaico. También aparecerá un paginador indicando el número de elementos encontrados, la página en la que nos encontramos y cuantas hay, facilitándonos así la navegación entre las distintas páginas.

4.2.3. Añadir o modificar sesión práctica

La siguiente interfaz se utilizará tanto para añadir una nueva sesión práctica como para modificarla. En este segundo caso el campo id estará desactivado, ya que ese valor nunca podrá ser modificado por un usuario. En caso de que sea necesario añadir una nueva asignatura dispondremos de un botón para poder llevar a cabo la acción, en el cual será necesario completar el formulario que aparece en la figura 16 (añadir nueva asignatura).

Figura 15. Interfaz para añadir una nueva sesión práctica.

4.2.4. Añadir o modificar asignatura

Al igual que en el caso anterior una misma interfaz nos servirá para realizar las dos acciones. Para añadir una nueva asignatura será necesario completar los siguientes campos: titulación, identificador, nombre, modulo, materia, curso y semestre. Tanto titulación como modulo y materia podrán añadirse nuevos desde esta interfaz. Para los dos segundos simplemente será añadiendo un nuevo elemento al combo para seleccionar. Mientras que en el caso de la titulación habrá que introducir el nombre de la titulación y su identificador.

Figura 16. Interfaz para añadir una nueva asignatura.

Capítulo 5

Implementación

En este capítulo quedará recogido todo lo referente a la implementación de la aplicación, desde la puesta en marcha del proyecto y requisitos previos hasta la finalización, destacando los aspectos más relevantes.

5. Implementación

Para llevar a cabo la implementación de la aplicación, me he ayudado de CakePHP, un framework *back-end* de desarrollo en PHP con el objetivo de que nos aporte ventajas a la hora del desarrollo, mantenimiento y seguridad. A continuación detallaré el proceso seguido de instalación y puesta a punto del proyecto con este framework.

5.1. CakePHP

5.1.1. Requisitos previos

Para la instalación del framework CakePHP, es necesario tener instalado previamente XAMPP y Composer.

XAMPP es un entorno de desarrollo que incluye Apache como servidor, MariaDB como base de datos SQL, PHP y Perl. Para instalarlo simplemente desde la página web oficial descargamos el instalador y lo ejecutamos. Una vez instalado XAMPP inicializamos los módulos Apache y SQL. Desde nuestro navegador accedemos a localhost para comprobar que se ha instalado y está funcionando correctamente.

Composer es un gestor de dependencias con el cual tendremos actualizadas las librerías que utilicemos y las versiones. Para instalarlo desde su página web, en el apartado download están las instrucciones para seguir paso a paso.

Para que CakePHP funcione correctamente en el servidor XAMPP es necesario modificar el fichero ini.php descomentando la siguiente línea:

```
extension=php_intl.dll
```

Una vez hecho esto tenemos que copiar todos los ficheros "ic*.dll" del directorio /xamp/php/ en el directorio bin de apache y reiniciar el servidor XAMPP.

5.1.2. Crear proyecto CakePHP

Una vez hemos realizado los anteriores pasos previos, pasamos a crear un proyecto con el framework CakePHP. Desde la web oficial del framework podemos encontrar la documentación.

```
composer create-project --prefer-dist cakephp/app [app-name]
```

En app-name escribiremos el nombre que queremos dar a nuestra aplicación.

Para comprobar que funciona correctamente nos dirigimos a la siguiente dirección desde nuestro navegador:

```
http://localhost/[app-name]/
```

Es necesario conectar la base de datos, para ello debemos modificarlo en el archivo *config/app.php*, introduciendo el apartado *Datasources* los datos de nuestra base de datos.

```
'Datasources' => [
        'default' => [
            'className' => 'Cake\Database\Connection',
            'driver' => 'Cake\Database\Driver\Mysql',
            'persistent' => false,
            'host' => 'localhost',

            //'port' => 'non_standard_port_number',
            'username' => 'root',
            'password' => '',
            'database' => 'gestordocumental',
            'encoding' => 'utf8',
            'timezone' => 'UTC',
            'flags' => [],
            'cacheMetadata' => true,
            'log' => false,
```

Figura 17. Configuración base de datos en CakePHP.

Debemos modificar lo señalado en amarillo, introduciendo los datos correctos para conectar con nuestra base de datos, nombre de usuario, contraseña y base de datos.
Una vez tenemos montado nuestro proyecto, toca estudiar su estructura de ficheros y peculiaridades para sacarle el máximo partido. Sobre esto y más cosas relacionadas con CakePHP hablaremos en el anexo 4.
Para que las fechas se almacenen de manera correcta, en el fichero Bootstrap.php es necesario incluir la siguiente sentencia:

```
date_default_timezone_set('Europe/Madrid');
```

5.2. Implementación de la base de datos

En un primer momento me dispuse a crear la base de datos en función a lo obtenido en los pasos previos de diseño, mediante el diagrama entidad relación y el modelo relacional, sin tener en cuenta el framework que iba a utilizar.
Esta elección no fue la correcta y se tuvo que modificar la base de datos ya que CakePHP, para agilizar el proceso de desarrollo, establece una serie de convenciones en los nombres de los distintos componentes de nuestra aplicación, de tal manera que más adelante nos facilite el desarrollo de código.
Las convenciones que son necesarias seguir y que por tanto hicieron necesario cambiar la implementación llevada a cabo son las siguientes:

- ✓ En primer lugar nombrar las tablas en inglés. Esto se debe a que más adelante usará este nombre en singular o plural, suprimiendo la "s" del final. El paso de plural a singular en castellano no siempre se resuelve igual, por lo que daría lugar a fallos.
- ✓ Los nombres de las tablas estarán en plural y si la tabla está formada por más de una palabra, separadas con guion bajo.
- ✓ Todas las tablas tienen que tener un identificador de tipo numérico autoincremental y que será la clave de la tabla. Este campo será controlado por CakePHP.
- ✓ Para hacer referencia a una clave foránea, será nombrada mediante el nombre de la tabla en singular seguido de *_id*.
- ✓ En el caso de unir varias tablas en relaciones varios a varios, el nombre de la tabla resultante no podrá ser aleatorio, sino que es necesario que aparezcan alfabéticamente ordenadas.

Por tanto mi base de datos ahora cumplía con las convenciones de CakePHP buscando así las ayudas facilitadas más adelante por el framework. Se puede observar en la siguiente figura, donde aparece el diseño adaptado al framework. El resto de convenciones quedan explicadas en el anexo 4.

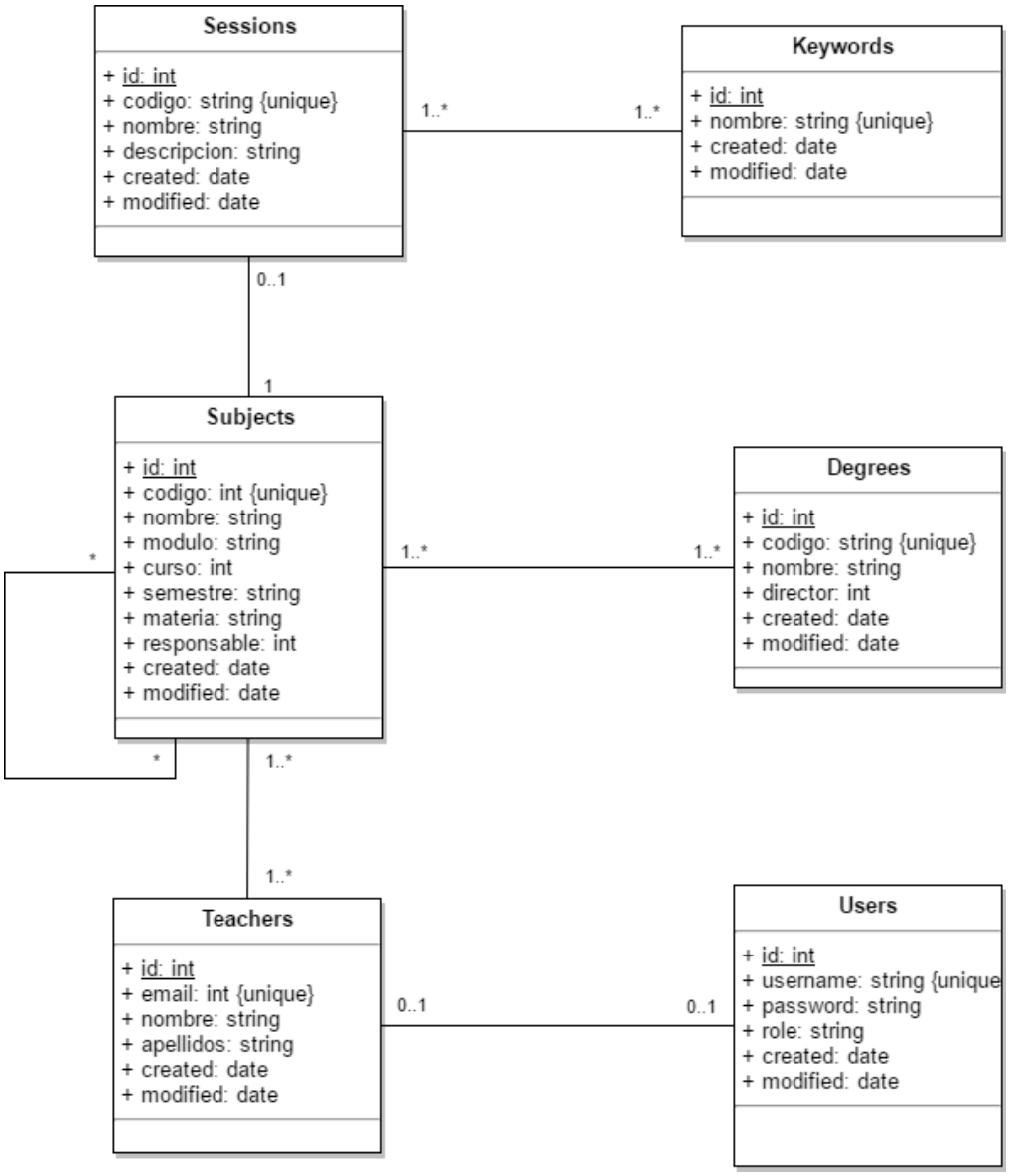


Figura 18. Diagrama clases adaptado al framework CakePHP.

Una vez realizadas todas estas modificaciones nos encontramos con el primer gran obstáculo en la realización del proyecto. Nos encontramos con una relación varios a varios reflexiva, en este caso entre las asignaturas. Siguiendo los pasos indicados en su web para llevar a cabo relaciones varios a varios me dispongo a realizar la relación. Para llevar a cabo esta relación, al ser contra el mismo modelo es necesario especificarlo a mano en el modelo definiendo las claves extrajeras. A pesar de seguir los pasos citados en la documentación oficial, mirar en foros y consultar numerosas páginas donde hayan realizado algo similar, no consigo que funcione la relación de la forma esperada. Tras demasiado tiempo invertido

en una tarea en la que no se estimaba tanto tiempo es necesario buscar una solución y poder continuar con el proyecto.
Se barajan 2 opciones para poder solventar este problema:

- ✓ Suprimir la relación entre las asignaturas, ya que para la funcionalidad de la aplicación no es relevante, recordemos que se quieren realizar búsquedas de sesiones filtrando por palabras clave y características de la asignatura, como módulo, curso, semestre, etc., por lo que la relación entre las asignaturas no interviene.
- ✓ La otra solución, y dado que no hay gran volumen de datos, era duplicar la tabla de las asignaturas y todos sus registros para realizar así la relación entre asignaturas.

Finalmente se opta por la primera opción, dado el tiempo limitado para implementar la segunda solución y el mínimo impacto que supone a la funcionalidad de la aplicación, puesto que para el buscador no interviene la relación entre las asignaturas.

5.2.1. Introducción de datos

Todos los datos que iba a manejar la aplicación, tanto de sesiones y palabras clave, como de asignaturas y profesores fueron facilitados por Judith Millán.
Para realizar la introducción de los datos contenidos en los ficheros, un Word y un Excel, se decidió procesarlos usando alguna librería que permitiera extraer toda la información necesaria de una manera más cómoda.
Se valoró el uso de varias librerías y en distintos lenguajes, pero finalmente se optó por la librería POI de Apache en Java. Se eligió esta librería ya que era un lenguaje en el que me sentía más cómoda y me resultaría más ágil realizarlo.
En el caso del fichero Word, con el contenido de las sesiones prácticas y sus palabras clave, se fue procesando párrafo a párrafo, buscando patrones para extraer los datos necesarios, como módulo, materia, asignatura, curso, semestre, nombre de la sesión, descripción y palabras clave asociadas. Estas dos últimas fueron marcadas antes con un testigo o marca *($ y &)* para poder tratarlas correctamente. A continuación un fragmento del fichero Word:

MÓDULO BÁSICO

MATERIA: BIOLOGÍA

CURSO: PRIMERO

SEMESTRE: PRIMERO

ASIGNATURA: 809-BIOLOGÍA

$ Microscopio compuesto

& Se aprende la estructura y el manejo de un microscopio compuesto (MC) mediante la observación de amiloplastos de patata y alubia. Para ello se realiza la preparación de amiloplastos de patata aplicando yodo-yoduro sobre un trozo de patata y comprobando el cambio de color a azul oscuro para a continuación realizar la preparación microscópica sobre el porta y con el cubre. Se realizan también preparaciones de amiloplastos de alubia sobre el porta y con el cubre a partir de semillas de alubia. Una vez realizadas las preparaciones microscópicas, se aprenden las normas de observación del MC y se aplican a la observación de las preparaciones de amiloplastos de patata y alubia.

Palabras clave: microscopio compuesto, preparación microscópica, porta, cubre, amiloplasto.

En el Excel se encontraban las asignaturas junto con los profesores asociados y responsables. Este se fue recorriendo fila a fila y procesando cada una de las columnas, extrayendo todos los datos necesarios de nada uno.
Para cada uno de los ficheros se generó un script SQL para introducir todos los datos obtenidos.

5.3. Implementación de la interfaz

Un aspecto muy importante de una aplicación web y que hay que tener en cuenta a la hora de desarrollarla es la apariencia. Para realizar una aplicación con una apariencia lo más atractiva posible y a la vez sencilla y amigable me he ayudado del framework *front-end* Bootstrap. A pesar de que la aplicación no está pensada para ser utilizada desde dispositivos móviles o pequeñas pantallas, un framework *front-end* puede ayudar a dar una buena apariencia a tu aplicación.

5.3.1. Integrar Bootstrap

Para integrar este framework lo que tenemos que hacer es descargar el plugin "BootstrapUI" con la ayuda de composer mediante el siguiente comando:

```
composer require friendsofcake/bootstrap-ui
```

Una vez descargado lo cargamos en nuestro proyecto mediante la siguiente sentencia:

```
bin/cake plugin load BootstrapUI
```

Por último, modificamos el fichero src/View/AppView.php añadiendo lo siguiente:

```
use BootstrapUI\View\UIViewTrait;

class AppView extends View
{
    use UIViewTrait;

    public function initialize()
    {
        $this->initializeUI(['layout' => false]);
    }
}
```

Figura 19. Inicialización de Bootstrap

Añadimos los ficheros de Bootstrap a nuestro proyecto, en el directorio webroot, que es donde están los ficheros css y js. Para finalizar modificamos las vistas con los estilos de Bootstrap.

```
<?= $this->Html->css('bootstrap.min') ?>
<?= $this->Html->script(['jquery-3.1.1', 'bootstrap.min']) ?>
```

El aspecto de la pantalla de inicio de sesión será el siguiente:

Figura 20. Pantalla inicio de sesión.

Por otro lado el aspecto del buscador para el rol superadministrador será el siguiente:

Figura 21. Pantalla principal de la aplicación (buscador).

Y la interfaz para añadir una nueva asignatura:

Figura 22. Pantalla para añadir una nueva asignatura.

5.4. Implementación del control de acceso

En un primer lugar la idea era que la aplicación se autenticara mediante el sistema de control de acceso de la universidad. Para ello nos pusimos en contacto con el servicio informático, para ponerles al tanto de lo qué queríamos hacer, que necesitábamos y si era posible. Tras varios correos electrónicos y una reunión en persona con los responsables del área encargada, acordamos que la mejor opción es que me facilitaran un web service, en el que a partir de la CUASI del profesor, me devolviera los datos que necesitaba (nombre, apellidos, email, si es profesor responsable de alguna asignatura y si es director de alguna titulación) y comprobaba si el login era correcto.
A falta de dos semanas para el depósito del proyecto, nos proporcionan un web service que devuelve en formato JSON los datos del profesor y las asignaturas a las que tiene acceso por ser profesor de la titulación y un valor booleano en función de si es responsable o no de las asignaturas. El web service no autentica y en ese momento me dicen que lo realice directamente contra active directory. Se solicitan los datos del active directory en ese mismo momento por correo electrónico, pero no fueron facilitados en tiempo para poder implementarlos.

5.4.1. Autenticación de usuarios

Lo primero que tenemos que hacer es añadir el componente de autenticación de usuarios. Para ello debemos añadir la siguiente línea en nuestro fichero AppController.php.

```
$this->loadComponent('Auth', [
            'authorize' => ['Controller'],
            'authenticate' => [
                'Form' => [
                    'fields' => [
                        'username' => 'username',
                        'password' => 'password'
                        ]
                    ]
                ],
                'loginAction' => [
                    'controller' => 'Users',
                    'action' => 'login'
                ],
                'authError' => 'Por favor, ingrese los datos de acceso.',
                'loginRedirect' => [
                    'controller' => 'Users',
                    'action' => 'home'
                ],
                'logoutRedirect' => [
                    'controller' => 'Users',
                    'action' => 'login'
                ],

            ]);
```

Figura 23. Componente de autenticación.

En el anterior fragmento de código lo que hacemos es cargar el componente de autorización de CakePHP y le damos valor a los parámetros de configuración. Especificamos que va ser un método de autorización por controlador (del que más adelante definiremos su comportamiento) y que vamos a autenticar mediante un formulario. En el apartado *fields* especificamos los campos de la base de datos de donde tomar el *username* y el *password*. Si hemos cumplido con las convenciones de CakePHP y en la tabla *Users* tenemos dos campos con estos nombres, los coge por defecto. En el apartado *loginAction* especificamos el controlador que va llevar acabo la autorización del usuario logeado y en *authError* un mensaje de error. Los apartados *loginRedirect* y *logoutRedirect* nos sirven para especificar a qué controlador y acción se los redirigirá después de realizar el login o logout.

En nuestro controlador UsersController tenemos que crear las acciones de login y logout para que el usuario pueda autentificarse y cerrar sesión.

```
public function login()
    {

        if ($this->request->is('post')) {
            $user = $this->Auth->identify();
            if ($user) {
                $this->Auth->setUser($user);
                return $this->redirect($this->Auth->redirectUrl());
            } else {
                $this->Flash->error('Usuario o contraseña incorrectos. Por
favor, inténtelo de nuevo.', [
                    'key' => 'auth'
                ]);
            }
        }
    }
```

Figura 24. Función login().

```
public function logout()
    {
        return $this->redirect($this->Auth->logout());
    }
```

Figura 25. Función logout().

También es necesario crear la vista login.ctp con el correspondiente formulario de login.

```
<div class="users form">
<?= $this->Flash->render('auth') ?>
<?= $this->Form->create() ?>
    <fieldset>
        <legend><?= __('Inicio de sesión') ?></legend>
        <?= $this->Form->input('username') ?>
        <?= $this->Form->input('password') ?>
    </fieldset>
<?= $this->Form->button(__('Entrar')); ?>
<?= $this->Form->end() ?>
</div>
```

Figura 26. Vista de login.ctp.

5.4.2. Autorización de usuarios.

Una vez definido nuestro componente de autenticación, en el cual hemos indicado que la autorización va ser mediante un controlador, es necesario definir el comportamiento de este en cada uno de los controladores mediante el método *isAuthorized($user)*.
En el caso del *AppController*, comprobamos si el usuario es superadministrador, para darle acceso a la aplicación al completo y en caso contrario no permitirles acceder.
En la siguiente figura aparece la función *isAutorized* del controlador *AppController*, donde damos total acceso a los usuarios superadministrador.

```
public function isAuthorized($user){
        if(isset($user['role'] && $user['role'] == 3')){
            return true;
        }
        return false;
    }
```

Figura 27. Función isAuthorized() para rol Súper administrador en AppController.

En cada controlador de nuestra aplicación debemos definir el método estableciendo los permisos determinados para cada usuario.

- ✓ Rol "Usuario básico": Podrá registrarse, iniciar sesión, modificar sus datos personales y realizar una consulta, lo que incluye seleccionar filtros e introducir palabras clave. También podrá visualizar todas las sesiones prácticas y palabras clave.
- ✓ Rol "Usuario avanzado": Además de lo anterior podrá añadir o modificar prácticas de la asignatura de la que sea responsable. También podrá añadir palabras clave.
- ✓ Rol "Administrador": Podrá realizar todo lo anterior además de añadir modificar asignaturas del plan del que sea director dicho usuario.
- ✓ Rol "Superadministrador": Podrá realizar todo lo anterior, además de añadir, modificar o eliminar titulaciones, añadir y eliminar usuarios o mandar una invitación de acceso a un usuario. Las operaciones de eliminar palabras clave, sesiones y asignaturas únicamente podrá realizarlas el súper administrador, ya que en un principio no estaba contemplado el eliminarlas.

Por ejemplo en el controlador de sesiones establecemos quién va tener acceso a cada acción, es decir qué roles pueden hacer determinadas operaciones. Tendríamos que controlar los roles “usuario básico”, “usuario avanzado” y “administrador”, ya que el usuario “superadministrador” podrá realizar cualquier operación.

```
//Administrador
if(isset($user['role']) && $user['role'] == 2){
   if(in_array($this->request->action, array('add', 'edit', 'index',
'view'))){
          return true;
            } else {
                if($this->Auth->user('id')){
                    $this->Flash->error(__('No tiene permisos de
acceso.'));
                    return $this->redirect([
                        'controller' => 'Users',
                        'action' => 'buscador'
                    ]);
                }
            }
        }
//usuario avanzado
if(isset($user['role']) && $user['role'] == 1){
   if(in_array($this->request->action, array('add', 'edit', 'index',
'view'))){
           return true;
            } else {
                if($this->Auth->user('id')){
                    $this->Flash->error(__('No tiene permisos de
acceso.'));
                    return $this->redirect([
                        'controller' => 'Users',
                        'action' => 'buscador'
                    ]);
                }
            }
        }
//usuario basico
if(isset($user['role']) && $user['role'] == 0){
   if(in_array($this->request->action, array('index', 'view'))){
          return true;
          } else {
                if($this->Auth->user('id')){
                    $this->Flash->error(__('No tiene permisos de
acceso.'));
                    return $this->redirect([
                        'controller' => 'Users',
                        'action' => 'buscador'
                    ]);

                }
                   }
        }
return parent::isAuthorized($user);
```

Figura 28. Función isAuthorized() de SessionsController.

Además de permitir acceso o no a diferentes puntos de nuestra aplicación, también será necesario modificar el menú con más o menos opciones en función del rol. Para ello la parte del menú será un fichero .ctp en el cual cargaremos más o menos opciones en función del rol. Este fichero luego podrá será cargado en cada una de las páginas donde sea necesario, evitando de esta manera tener que escribir el mismo código en distintos ficheros. Para realizar esto nos ayudamos de la funcionalidad Element que nos proporciona CakePHP. Todos los ficheros situados en el directorio template/element podrán ser cargados con la siguiente instrucción:

```
<?= $this->element('cabecera_menu', array("role" =>
$current_user['role'])) ?>
```

Para no tener que cargar este contenido en todas las páginas de nuestra aplicación, lo cargamos en la vista default.ctp, que es común a todas y con realizarlo una única vez será suficiente. A continuación un fragmento del código utilizado para el menú del rol "superadministrador".

```
<div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
      <ul class="nav navbar-nav">
        <li><a href="#"> Realizar Consulta </a></li>
        <li class="dropdown">
              <a href="#" class="dropdown-toggle" data-toggle="dropdown"
role="button" aria-haspopup="true" aria-expanded="false">Palabras Clave <span
class="caret"></span></a>
            <ul class="dropdown-menu">
            <li><?= $this->Html->link('Añadir palabra clave' ,
                    array('controller' => 'keywords', 'action' => 'add'))?></li>
            <li role="separator" class="divider"></li>
            <li><?= $this->Html->link('Ver Todas' ,
                    array('controller' => 'keywords', 'action' => 'index'))?></li>
          </ul>
              </li>
        <li class="dropdown">
            <a href="#" class="dropdown-toggle" data-toggle="dropdown"
role="button" aria-haspopup="true" aria-expanded="false">Sesiones Prácticas <span
class="caret"></span></a>
            <ul class="dropdown-menu">
            <li><?= $this->Html->link('Añadir sesión práctica' ,
                    array('controller' => 'sessions', 'action' => 'add'))?></li>
            <li role="separator" class="divider"></li>
            <li><?= $this->Html->link('Ver Todas' ,
                    array('controller' => 'sessions', 'action' => 'index'))?></li>
          </ul>
        </li>
        <li class="dropdown">
            <a href="#" class="dropdown-toggle" data-toggle="dropdown"
role="button" aria-haspopup="true" aria-expanded="false">Asignaturas <span
class="caret"></span></a>
            <ul class="dropdown-menu">
            <li><?= $this->Html->link('Añadir Asignatura' ,
                    array('controller' => 'subjects', 'action' => 'add'))?></li>
            <li role="separator" class="divider"></li>
            <li><?= $this->Html->link('Ver Todas' ,
                    array('controller' => 'subjects', 'action' => 'index'))?></li>
          </ul>
        </li>
      </ul>
```

Figura 29. Fragmento menú de acciones para rol Administrador.

5.5. Implementación de funcionalidad de la aplicación

5.5.1. Buscador de sesiones

La funcionalidad que se espera del buscador de sesiones es poderlas filtrar por las palabras clave con las que cada sesión esta etiquetada y también por las características de la asignatura a la que pertenecen, por ejemplo, modulo, curso o semestre.
Nos podemos encontrar con 4 casos, el primero en el que no nos introducen ningún filtro ni palabra clave, en el que mostraremos todas las sesiones.
Un segundo caso en el que nos introducen el filtro de las asignaturas. Para tratar este caso lo primero que necesitamos es saber qué asignaturas cumplen con el filtro fijado, para buscar sesiones prácticas de dichas asignaturas. Una vez tenemos las asignaturas que cumplen las características del filtro, obtenemos las sesiones que tienen y las mostramos.
Un tercer caso es en el que nos introducen únicamente palabras clave. Para obtener las sesiones que mostrar necesitaremos realizar una búsqueda que nos devuelva las sesiones etiquetadas con esas palabras clave.
Y por último el caso en el que nos introduzcan los dos filtros de búsqueda, el filtro de asignaturas y el filtro de palabras clave. En este caso es necesario comprobar las sesiones por dos caminos. El primero por las asignaturas que cumplen el filtro, el otro por el de las palabras clave y realizar la unión de ambas.
Para poder obtener las sesiones correctamente es necesario procesar todos los parámetros de muestro formulario en un controlador. Para ello creamos un formulario, el cual contendrá el filtro y el input de palabras clave, indicando el método y que controlador va ser quien lo trate:

```
<?= $this->Form->create('Session', array ('type' => 'POST', 'class' =>
'form-search', 'url' => array('controller' => 'users', 'action' =>
buscador))); ?>
```

Figura 30. Formulario buscador sesiones.

En el controlador lo que tenemos que hacer es recoger todos los parámetros del filtro, obtener su contenido para así saber en cuál de los 4 casos explicados anteriormente nos encontramos. Una vez tengamos esos valores podremos realizar las consultas adecuadas.

Para realizar una consulta el framework CakePHP utiliza una nomenclatura propia sustituyendo al tradicional SQL. Por ejemplo para obtener todas las asignaturas la consulta seria la siguiente:

```
$asignaturas = $asig->find('all', array('fields' => array('Subjects.id'),
'conditions' => array($conditions)))->toArray();
```

Figura 31. Consulta obtención asignaturas.

Donde fields sería el select de una consulta sql y conditions el where. También podemos añadir otro parámetro 'order' que sería un típico order by de las consultas sql tradicionales. La variable conditions puede ser una única condición o cláusula o varias. En el caso de ser un array de condiciones estas por defecto estarían unidas por el operador AND. Si se quisieran unir por el operador OR la consulta quedaría de la siguiente manera.

```
$asignaturas = $asig->find('all', array('fields' => array('Subjects.id'),
'conditions' => array('OR' => $conditions)))->toArray();
```

Figura 32. Consulta obtención asignaturas con condiciones.

5.5.2. Autocompletado de palabras clave en el buscador

Una utilidad que en este caso es muy importante es el autocompletado de palabras a la hora de realizar una búsqueda, ya que son muchas las palabras posibles y no muy comunes, por lo que esta utilidad facilita bastante la búsqueda al usuario.
Para ello voy a utilizar la librería jQueryUI, mediante una librería personalizada desde la página oficial (https://jqueryui.com/). La configuración de los componentes a utilizar ha sido la siguiente:

- ✓ UI Core completo.
- ✓ Interactions completo.
- ✓ Autocomplete widget.
- ✓ Menu widget.
- ✓ Theme Smoothness.

Una vez descargada nuestra librería personalizada, debemos añadir los ficheros *js* y *css* a nuestro proyecto en las capetas *js* y *css* de nuestro directorio *webroot* y añadirlas a los ficheros del proyecto. Ahora necesitamos crear una función javascript para poder llevar a cabo la función de autocompletar las palabras clave.

```
$(document).ready(function(){
    $("#s").autocomplete({
        minLength: 2,
        select: function(event, ui){
            $("#s").val(ui.item.label);
        },
        source: function(request, response){
             $.ajax({
                url: basePath + "keywords/searchJSON",
                data: {
                    term: request.term
                },
                dataType: "json",
                success: function(data){
                    response($.map(data, function(el, index){
                        return {
                            value: el.nombre,
                            nombre: el.nombre
                        };
                    }));
                }
             });
        }
    }).data("ui-autocomplete")._renderItem = function(ul, item){
        return $("<li></li>")
        .data("item.autocomplete", item)
        .append("<a>" + item.nombre + "</a>")
        .appendTo(ul)
    };
});
```

Figura 33. Función JS de autocompletado.

Después en nuestro controlador de palabras clave, *keywordsController.php*, tenemos que realizar la función que nos va devolver la lista de palabras en función de lo que vayamos introduciendo en nuestro *input*.

```
public function searchJSON(){
	$term = null;
	if(!empty($this->request->query['term']))
		{
		$term = $this->request->query['term'];
		//palabras separadas por espacio
		$terms = explode(' ', trim($term));
		$terms = array_diff($terms, array(''));
		foreach ($terms as $term) {
			$conditions[] = array('Keywords.nombre LIKE' => '%' .
$term . '%');
		}

		$keyW = $this->Keywords->find('all', array('recursive' => -1,
'fields' => array('Keywords.nombre'), 'conditions' => $conditions, 'limit'
=> 20))->toArray();
	}
	$serialized = serialize($keyW);
	echo json_encode($keyW);
	$this->autoRender = false;
}
```

Figura 34. Función obtención de palabras coincidentes.

De esta manera estamos devolviendo las palabras que contengan el texto que estamos escribiendo en el input.

5.5.3. Añadiendo prácticas

Para añadir una práctica a nuestra aplicación es necesario rellenar un formulario con los datos de una sesión práctica, como son nombre, descripción y palabras clave asociadas. Para añadir palabras a nuestra sesión las iremos seleccionando desde un *listbox* que aparecerá en la interfaz. Si fuera necesario añadir una palabra clave podría realizarse tanto antes como después, bien indicando en la creación de la palabra clave la sesión en la que aparece o añadiéndola posteriormente a la sesión. Este formulario contara con las útiles y necesarias reglas de validación así como indicaciones de lo que se espera en cada campo mediante *placeholders*. También se ha implementado una función javascript en determinados *input* para forzar la introducción en mayúsculas, siguiendo así con el formato de los datos introducidos desde el fichero.

5.5.4. Añadiendo asignaturas

Al añadir una asignatura ocurre igual que en el caso anterior, se presenta un formulario el cual hay que rellenar correctamente para poder añadir una asignatura. Además al añadir una asignatura será necesario rellenar el profesor responsable de la misma y los profesores que la imparten si fuera más de un profesor y pudiendo ser más de uno. También habrá que seleccionar la titulación a la que pertenece la asignatura, pudiendo ser más de una también. Según el rol del usuario que se disponga a añadir una asignatura, la lista de titulaciones variará, mostrando únicamente la que es director, para el rol administrador o todas para el rol superadministrador.

5.5.5. Añadiendo usuarios

Para añadir un usuario como en el caso anterior será necesario rellenar un formulario. En este formulario será necesario añadir un profesor entre una lista de los profesores que aún no tienen usuario asignado. Por defecto este usuario inicialmente tendrá el rol usuario básico, que solo el usuario con rol superadministrador podrá modificar otorgándole mayores privilegios. La lista de profesores disponibles se irá actualizando para que no sea posible encontrarnos con 2 usuarios asignados a un mismo profesor.
Otro aspecto importante y a tener en cuenta de la gestión de usuarios es el almacenamiento de las contraseñas en nuestra base de datos. Por un aspecto de seguridad estas contraseñas no son mostradas nunca en claro, ni tampoco almacenadas. Para almacenarlas me he ayudado de una utilidad del framework para hasearlas, de manera que se almacenarán cifradas.

5.5.6. Eliminando datos de nuestra aplicación

Dado que en un inicio esta operación no estaba contemplada ya que se querían almacenar todos los datos a modo de repositorio o almacén, solo el usuario con rol superadministrador podrá llevarla a cabo. Antes de cualquier eliminación la aplicación mostrará un mensaje de confirmación mostrando información del dato a eliminar para que el usuario confirme o cancele.
Cada una de las operaciones a realizar por cada rol vendrá detalladas en un manual de usuario, que se facilitará a los usuarios finales de la aplicación (anexo 6).

5.6. Controlando los errores

El control de errores es un punto muy importante de toda aplicación web, y que es necesario controlar de la forma adecuada. En mi caso no he considerado necesario mostrar información acerca del error al usuario en los casos en los que la url no estaba bien escrita, desembocando así en un error el cual no era capaz de mostrar ninguna página de la aplicación.
Para evitar este comportamiento he introducido en el controlador general de la aplicación una función beforeRender, que como su propio nombre indica se ejecutará antes de renderizar cualquier página. En esta función lo que hacemos es capturar el error y redirigir nuestra aplicación a la página principal.

```
public function beforeRender(Event $event)
{
 if ($this->response->statusCode() == '404'
       || $this->response->statusCode() == '400' ) {
          return $this->redirect([
                        'controller' => 'Users',
                        'action' => 'buscador']);
    }
}
```

Figura 35. Función de control de errores.

En los casos en los que los errores sean al añadir eliminar o modificar alguno de los elementos de la aplicación como asignaturas, titulaciones, sesiones, etc. se mostrará un mensaje informativo indicando que se ha producido un error y la operación no se ha llevado a cabo correctamente.

5.7. Pruebas

Debido a los problemas encontrados al comienzo de la fase de implementación del proyecto, las pruebas se han ido llevando a cabo durante todo el desarrollo del proyecto. Estas pruebas consistían tanto en pruebas funcionales de las nuevas características implementadas, como del diseño, que fuese lo más cercano a la realidad. En una última fase de pruebas final, la aplicación ha sido probada por dos agentes externos del proyecto y con diferentes niveles en conocimientos informáticos. Además se ha redactado un manual de usuario en el cual se detallan todas las funcionalidades de la aplicación y sus pasos para realizarlas correctamente.

Capítulo 6

Despliegue en el servidor

En este capítulo se explicará el proceso llevado a cabo para poner en producción la aplicación en el servidor de la universidad.

6. Despliegue en el servidor

Otro punto importante del proyecto era poder desplegar la aplicación en un servidor de la Universidad, para que así los profesores del grado interesados en el gestor pudiesen acceder. Tras cerca de un mes de tramites de solicitud del servidor virtual nos aceptan la petición y proporcionan los datos de la máquina. Por temas de seguridad, esta se encuentra en una zona protegida y que por tanto no es accesible desde fuera de la red de la Universidad. Para poder acceder desde fuera necesario establecer una VPN entre mi equipo y la red de la universidad, lo que supone otro formulario de solicitud. Una vez nos dan acceso a la maquina ya puedo trabajar con ella desde mi equipo en casa.

Para alojar nuestra aplicación en host proporcionado por el servicio informático, fue necesario realizar varias configuraciones en la máquina, como por ejemplo instalar phphmyadmin, para gestionar la base de datos o un ftp para poder subir los ficheros necesarios. Estos pasos quedan detallados en el anexo 5, instalación server LAMP.

Siguiendo las instrucciones del framework para realizar el despliegue en el servidor apache, fue necesario realizar las siguientes configuraciones.

En primer lugar modificar el fichero el fichero de configuración de apache (apache.conf) permitiendo la sobrescritura modificando las siguientes líneas:

```
<Directory />
    Options FollowSymLinks
    AllowOverride All
</Directory>
<Directory /var/www>
    Options Indexes FollowSymLinks MultiViews
    AllowOverride All
    Order Allow,Deny
    Allow from all
</Directory>
```

En el fichero .htacces de la raíz de nuestro proyecto Cakephp es necesario realizar la siguiente configuración:

```
<IfModule mod_rewrite.c>
   RewriteEngine on
   RewriteRule    ^$    webroot/    [L]
   RewriteRule    (.*) webroot/$1    [L]
</IfModule>
```

Y el fichero .htacces del directorio webroot de nuestro proyecto debe quedar asi:

```
<IfModule mod_rewrite.c>
    RewriteEngine On
    RewriteCond %{REQUEST_FILENAME} !-f
    RewriteRule ^ index.php [L]
</IfModule>
```

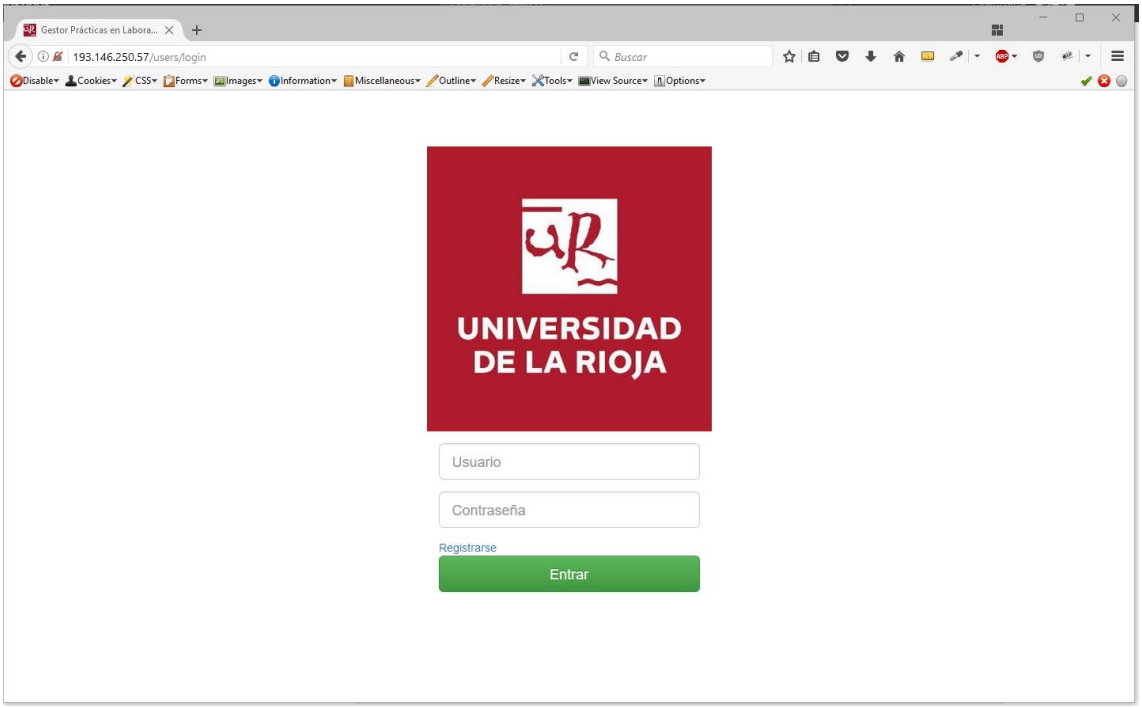
Además de esto es necesario tener habilitada la extensión intl. Para ello ejecutamos el siguiente comando:

```
apt-get install intl-php7
```

Y también el módulo de sobreescritura:

```
a2enmod rewrite
```

Reiniciamos el servidor apache para que se ejecuten los cambios realizados, y podemos comprobar como nuestra aplicación ha sido correctamente desplegada en el servidor.



Capítulo 7

Seguimiento y control

Este capítulo se reflejarán diferentes aspectos sobre el seguimiento y control del proyecto llevado a cabo durante la realización del mismo, así como objetivos cumplidos y una tabla comparativa de horas estimadas y reales.

7. Seguimiento y control

7.1. Objetivos alcanzados

Los objetivos fundamentales del proyecto se han cumplido satisfactoriamente. La aplicación permite buscar sesiones en función de un filtro, por palabras clave o ambos. Además la aplicación permite añadir nuevas palabras clave, sesiones, asignaturas y titulaciones. También editarlas y eliminarlas. Establecer nuevas relaciones entre las asignaturas ya existentes y sesiones, sesiones con palabras clave y también eliminar relaciones anteriores. Por otro lado la aplicación permite a los usuarios registrarse, iniciar sesión y realizar diferentes operación en función del rol que tengan asignado.

7.2. Objetivos no alcanzados

Los objetivos no alcanzados principalmente han sido en los que he colaborado con el servicio informático, como la de autenticación mediante el sistema de la Universidad de La Rioja, o la obtención de datos de los profesores.
La causa principal del fracaso de estos objetivos ha sido la falta de tiempo. Al intervenir el servicio informático los tiempos de espera en la obtención de información, datos necesarios y servicios se ha visto afectada.
Esto no ha causado un gran impacto sobre la funcionalidad de la aplicación puesto que se ha establecido un sistema de autenticación de usuario de forma local.

7.3. Tabla comparativa horas estimadas/reales

En la siguiente tabla se puede ver una comparativa entre los tiempos estimados y reales, junto con los motivos de sus desviación para cada una de las tareas llevabas a cabo en el proyecto.

Código	Tarea	Horas estimadas	Horas reales	Motivos de la desviación
1	Planificación	10 h	12 h	Se necesitó mayor tiempo del esperado.
2	Gestión	55 h	50 h	-
2.1	Elaboración de la memoria	30 h	35 h	Mayor tiempo del estimado en la redacción, por los cambios durante el proyecto.
2.2	Seguimiento y control	20 h	15 h	Reuniones de menor duración a lo esperado.
2.3	Defensa	15 h	-	La preparación de la defensa no se ha realizado a fecha de depósito del proyecto
3	Documentación y formación	45 h	60 h	-
3.1	Estudio frameworks	15 h	15 h	-

3.2	Documentación	30 h	45 h	Dado los problemas encontrados he necesitado más tiempo de documentación del esperado
4	Análisis	10 h	9 h	
4.1	Requisitos	4 h	3 h	Estaban bastante claros e intuitivos desde un primer momento, por lo que no fue necesario emplear demasiado tiempo.
4.2	Diagrama casos de uso y de actividad	6 h	6 h	-
5	Diseño	10 h	12 h	-
5.1	Diagrama de clases	4 h	4 h	-
5.2	Base de datos	4 h	6 h	Dado la modificación necesaria durante el proyecto.
5.3	Interfaz	2 h	2 h	-
6	Implementación	165 h	175 h	
6.1	Base de datos	30 h	40 h	Problemas encontrados por convenciones y relación reflexiva.
6.2	Introducción datos desde fichero	15 h	25 h	Más tiempo del esperado por la modificación de la base de datos.
6.3	Interfaz Web	55 h	45 h	CakePHP ayuda bastante y el uso de Bootstrap no me ha resultado dificultoso. He tenido que priorizar la funcionalidad ante el aspecto.
6.4	Lógica de la aplicación	65 h	60 h	CakePHP facilita muchas de las operaciones básicas, pero por otro lado tiene peculiaridades que hacen lento el desarrollo.
7	Pruebas	15 h	5 h	La mayor parte de las pruebas se llevaron a cabo durante el desarrollo del proyecto, por lo que solo se dedicaron 5 h al finalizarlo.
	Total	**310 h**	**318 h (*)**	

Figura 36. Tabla comparativa de tiempos.

(*) Faltarían de añadir las 15 h estimadas en la preparación de la defensa que no se han llevado a cabo antes de la fecha de depósito del proyecto. Lo que sumarian un total de 333 horas.

En conclusión podemos considerar que 8 horas de desviación, cerca de un 3%, en un proyecto de 300 horas es una desviación aceptable. Esta desviación tan pequeña se ha podido lograr gracias a contar con tiempo disponible desde el principio de la planificación.

Capítulo 8

Conclusiones

En este capítulo quedarán recogidas las conclusiones y reflexiones tras la realización del proyecto, así como las posibles mejoras de la aplicación en un futuro

8. Conclusiones

En primer lugar, este proyecto me ha servido para darme cuenta de todos los conocimientos que a lo largo de estos años de estudios he ido adquiriendo. Este proyecto para mí era un reto personal, ya que afortunadamente o no, decidí realizar la aplicación en un lenguaje que nunca antes había visto. Esto supuso en un primer lugar mucho miedo e incertidumbre, pero a medida que avanzaba en el desarrollo comprendía que una vez adquieres los conocimientos básicos, lo básico es siempre igual. Evidentemente cada lenguaje tiene sus peculiaridades y hay que conocerlas y tenerlas en cuenta.

Por otro lado, me ocurrió lo mismo al trabajar con un framework. En un primer momento parece que va ayudar mucho, pero también son en otros aspectos en los que te pondrá en más de un apuro, quizás por mi inexperiencia en el uso de ellos o el tiempo ajustado del que disponía.

Este era el primer proyecto real y de una envergadura considerable al que me enfrentaba sola.

Al intervenir una tercera parte como ha sido el servicio informático, me ha hecho darme cuenta de las desviaciones que puede causar en un proyecto real si no se estiman adecuadamente los tiempos y coordinan las diferentes actividades.

En mi caso, al solicitar información sobre el acceso de los usuarios a las aplicaciones corporativas, es un tema que se trató con mucha cautela, no dándome acceso total fácilmente y restringiendo lo máximo posible el acceso desde fuera de la universidad, lo que no me permitió dedicarle todo el tiempo que hubiera requerido.

Ante un primer proyecto de gran envergadura al que me enfrentaba sola, únicamente con la ayuda de mi tutor Ángel Luis, puedo estar satisfecha de los resultados obtenidos y la capacidad de solventar los obstáculos encontrados a lo largo del desarrollo.

- ✓ Posibles mejoras

Como posibles mejoras de la aplicación se podría valorar hacerla responsiva para dispositivos más pequeños, como tablets, ya que hoy en día estos dispositivos se usan cada vez más.

También si durante el uso de la aplicación se viera que es necesario o interesante tener las relaciones entre las asignaturas se podría implementar, bien como un atributo más de la asignatura en el que se guarden las asignaturas con las que tiene relación, o duplicando la tabla para realizar la unión entre dos entidades.

Por otro lado sería una mejora a tener encuentra integrar el sistema de autenticación de la Universidad de La Rioja que finalmente no pudo llevarse a cabo.

Capítulo 9

Bibliografía

En este capítulo quedarán recogidas las principales referencias utilizadas y consultadas durante la realización del proyecto.

9. Bibliografía

I. CakePHP documentación oficial
https://book.cakephp.org/3.0/en/index.html
https://api.cakephp.org/3.3/

II. Foros para programadores
http://stackoverflow.com/
http://www.forosdelweb.com

III. Librerías utilizadas
https://jquery.com/
http://jqueryui.com/

IV. Bootstrap
http://getbootstrap.com/

V. Documentación PHP
https://secure.php.net/manual/es/index.php

VI. Otras páginas web de ayuda
http://librosweb.es/
https://desarrolloweb.com/
https://www.tutorialspoint.com/